FADE, DISSOLVE AND WIPE PRODUCTION IN MPEG-2 COMPRESSED VIDEO

W. A. C. Fernando, C. N. Canagarajah and D. R. Bull
Image Communications Group, Centre for Communications Research, University of Bristol,
Merchant Ventures Building, Woodland Road, Bristol BS8 1UB, United Kingdom
E-mail: W.A.C.Fernando@bristol.ac.uk

ABSTRACT

With the increase of digital technology in video production, several types of complex video special effects editing have begun to appear in video clips. In this paper a novel algorithm is proposed for fade-in, fade-out, dissolve and wipe video special effects editing in compressed video without full frame decompression and motion estimation. DCT coefficients are estimated and use these coefficients together with the existing motion vectors for these special effects editing in compressed domain. Results show that both objective and subjective quality of the edited video in compressed domain closely follows the quality of the edited video in uncompressed video at the same bit rate.

1. INTRODUCTION

Due to the increase in video products such as digital cameras, camcorders and storage devices (DVDs), digital video editing is becoming increasingly popular. Video editing effects are needed to enhance the quality of the video production. Most video editing can be divided into two major categories: abrupt transitions and gradual transitions. Gradual transitions include camera movements: panning, tilting, zooming and video editing special effects: fade-in, fade-out, dissolving, wiping. Abrupt transition is the simplest edit between two shots in which the transition is immediate between two frames. Special effects occur gradually over multiple frames. Though, the number of possible video special effects is quite high in video production, most of these special effects fall into fading, dissolving or wiping. During a fade, the intensity gradually decreases to, or increases from, a solid colour. In a dissolve, two shots, one increasing in intensity, and the other decreasing intensity, are additively mixed. Wipes are generated by translating a line across the frame in some direction, where the content on the two sides of the line belong to the two shots separated by the edit. All these special effects are used to produce gradual transitions between two scenes. These video editing tools are designed for spatial domain processing.

The large channel bandwidth and memory requirements for the transmission and storage of image and video necessitate the use of video compression techniques [1,2,3]. Hence, the visual data in multimedia databases is expected to be stored mostly in the compressed form. Thus editing of compressed video is also essential. Therefore, a typical desktop video editing system must first convert the compressed domain representation to a spatial domain representation and then perform the editing function on the spatial domain data. Then, output of the editing system must be recompressed. This increases the overall computational complexity of the editing process. In order to avoid the unnecessary decompression operations and compression processes, it is efficient to edit the image and video in the compressed format itself.

Though video editing in spatial domain is well established, work on processing video effects in compressed domain has not received a lot of attention. Since, Discrete Cosine Transform (DCT) is chosen as the basis function in compressions standards [1, 2, 3], most of the compressed-domain based processing methods were developed to deal with DCT domain processing of the data. The basic idea in DCT-domain based processing is that the spatial domain processing can be replaced by an equivalent processing of the DCT-domain representations. Therefore, most previous approaches on video editing in compressed domain have been performed in DCT domain. Previous approaches can be divided into two categories: special effects editing for JPEG images and special effects for MPEG sequences. Shen *et al* [4] proposed inner-block transform method to perform regular geometric transformations for JPEG images. Shen *et al* also developed fast algorithms for DCT domain convolution [5]. It was shown that pixel-wise multiplication in the spatial-domain can be replaced by a convolution function in the DCT domain. Smith *et al* [6] showed how the algebraic operation of pixel-wise and scalar addition and multiplication, can be done in DCT compressed domain. Authors used these operations in JPEG images to implement two common video transformations: dissolving and sub-titling. They argued that these scalar addition and multiplication could be implemented on quantised DCT coefficients. However due to the non-linear behaviour of the mapping function, many problems were introduced with this scheme [6]. All these DCT domain based approaches have been applied only for JPEG images. Shen have proposed DC-only fade-out operation for MPEG compressed video [7]. This algorithm is proposed under the assumption that fade-out is viewed as a reduction of picture brightness. However, this is a poor assumption of the actual fade-out operation in video production.

Original manuscript received June 19, 2000

Currently, most existing video processing operations performed on the compressed sequences require motion estimation or full frame decompression. To maximise the benefits from data compression without incurring extra computation and storage, it would be advantageous to develop processing algorithms that do not require decompression of the entire compressed data. Operations on compressed bit streams directly or with minimal decoding of relevant information will then eliminate the computational time necessary for full decompression and the extra storage needed for the decompressed results. In this paper we present a novel technique for fade-in, fade-out, dissolve and wipe editing in compressed video without full frame decompression and re-compression.

Rest of the paper is organised as follows. In section 2 mathematical models for fading, dissolving and wiping are presented. Section 3 discusses an overview for MPEG-2 video, a model for uncompressed video editor and the conventional compressed domain video editor. It also describes the process of DCT coefficients extraction from compressed domain. Proposed scheme for video special effects editing is presented in section 4. Some experimental results are given in section 5. Finally, section 6 presents the conclusions and future work.

2. MATHEMATICAL MODEL FOR VIDEO EDITING

In this section mathematical models for fade-in, fade-out, dissolving and wiping are discussed.

2.1 Dissolving/Fading

In video editing and production, proportions of two or more picture signals are simply added together so that the two pictures appear to merge on the output screen. Very often this process is used to move on from picture F to picture G. In this case, the proportions of the two signals are such that as the contribution of picture F changes from 100% to zero, the contribution of picture G changes from zero to 100%. This is called dissolving. When picture F is a solid color, it is called as fade-in and when picture G is a solid colour, it is known as fade-out. Therefore, dissolving, fade-in and fade-out can be modelled as shown in Equations (1), Equation (2) and Equation (3) respectively.

$$s_n(x,y)=\begin{cases} f_n(x,y) & 0\le n<L_1 \\ \left[1-\left(\frac{n-L_1}{L}\right)\right]f_n(x,y)+\left(\frac{n-L_1}{L}\right)g_n(x,y) & L_1\le n\le (L_1+L) \\ g_n(x,y) & (L_1+L)<n\le L_2 \end{cases} \quad (1)$$

$$s_n(x,y)=\begin{cases} f_n(x,y) & 0\le n<L_1 \\ \left[1-\left(\frac{n-L_1}{L}\right)\right]C+\left(\frac{n-L_1}{L}\right)g_n(x,y) & L_1\le n\le (L_1+L) \\ g_n(x,y) & (L_1+L)<n\le L_2 \end{cases} \quad (2)$$

$$s_n(x,y)=\begin{cases} f_n(x,y) & 0\le n<L_1 \\ \left[1-\left(\frac{n-L_1}{L}\right)\right]f_n(x,y)+\left(\frac{n-L_1}{L}\right)C & L_1\le n\le (L_1+L) \\ g_n(x,y) & (L_1+L)<n\le L_2 \end{cases} \quad (3)$$

where, C is the video signal level (solid colour), $s_n(x,y)$ is the resultant video signal, $f_n(x,y)$ is picture F, $g_n(x,y)$ is picture G, L_1 is length of sequence F, n is the frame number, L is length of dissolving/fading sequence and L_2 is length of the total sequence.

2.2 Wiping

Wiping is a transition from one scene to another wherein the new scene is revealed by a moving boundary. This moving boundary can be any geometric shape. However in practice this geometric shape is either a line or a set of lines. According to the geometric shape of this boundary, there are about 20-30 different moving boundaries used for wiping in video production. Wiping can be modelled as shown in Equation (4).

$$s_n(x,y)=\begin{cases} f_n(x,y) & n<L_1 \\ P_n\otimes f_n+\overline{P}_n\otimes g_n & L_1\le n\le (L+L_1) \\ g_n(x,y) & (L+L_1)<n\le L_2 \end{cases} \quad (4)$$

where, "$\otimes$" denotes element by element matrix multiplication and matrix P generates the wiping pattern, P_n matrix represent the wiping transition (elements of P_n are either "1" or "0" always).

3. VIDEO EDITING IN COMPRESSED DOMAIN

3.1 MPEG-2 overview

MPEG-2 video compression is used in many current and emerging products. The basic idea behind MPEG video compression is to remove spatial redundancy within a video frame and temporal redundancy between video frames. As in JPEG, the standard for still image compression, DCT-based (Discrete Cosine Transform) compression is used to reduce spatial redundancy. Motion- compensation is used to exploit temporal redundancy.

MPEG-2 video is broken up into a hierarchy of layers to help with error handling, random search editing, and synchronisation. From the top level, the first layer is known as the video sequence layer. The second layer down is the group of pictures (GOP), which is composed of one or more groups of intra (I) frames and/or non-intra (P and/or B) frames. The third layer down is the picture layer itself, and the next layer beneath is called the slice layer. Each slice consists of macroblocks (MBs), which are 16x16 arrays of

luminance pixels, or picture data elements, with 8x8 arrays of associated chrominance pixels. The MBs can further be divided into 8x8 blocks, for further processing.

MPEG-2 picture types

MPEG-2 defines three types of pictures: Intra-pictures (I-Pictures), Predictive pictures (P-Pictures) and Bi-directional pictures (B-Pictures).

Intra-pictures (I-pictures): These pictures are encoded only with respect to themselves. Here each picture is decomposed into blocks of 8x8 pixels each and encoded directly using DCT transformation process.

Predictive pictures (P-pictures): These are pictures encoded using motion compensated prediction from a past I/P picture. A prediction error is calculated between a 16x16 pixels region (MB) in the current picture and the past reference I/P picture. A motion vector is also calculated to determine the value and direction of the prediction. The prediction error is transmitted after DCT coding.

Bi-directional pictures (B-pictures): These are pictures encoded using motion compensation prediction from a past and/or next I/P picture. A prediction error is calculated between a 16x16 pixels region in the current picture and the past as well as next reference I/P picture. Two motion vectors are also calculated to determine the value and direction of the prediction from both directions.

Figure 1 shows a typical MPEG-2 video sequence with GOP of 12 (M=12) and sub-GOP size of 3 (N=3).

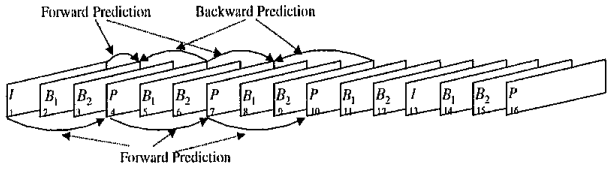


Figure 1: A Typical MPEG-2 Compressed Video Sequence

3.2 Uncompressed domain video editing

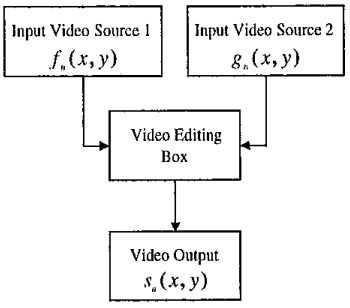


Figure 2: Uncompressed domain video editor

Figure 2 illustrates an uncompressed domain video editor. The function of the editing box is to process two input video signals to produce fading, dissolving or wiping according to the mathematical model explained in section 2. When both $f_n(x,y)$ and $g_n(x,y)$ are compressed, this operation cannot be done in compressed domain without some additional processing. A straightforward way to performing these operations on compressed sequence would be to decompress the sequences, apply operations in the spatial domain and recompress the video output. Figure 3 presents this conventional video editing model. However the costly IDCT, DCT and motion estimation operations make it difficult for real time applications.

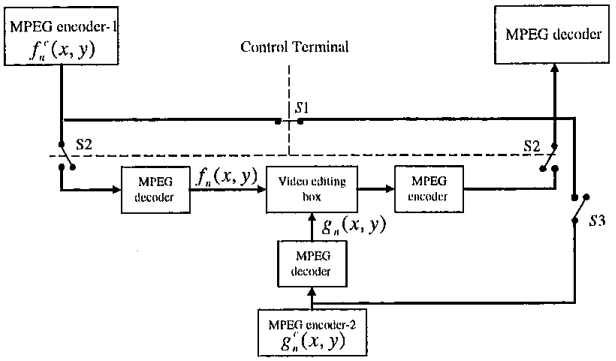


Figure 3: Conventional compressed domain video editor
(f_n^c – Compressed signal of f_n and g_n^c – Compressed signal of g_n)

3.3 DCT coefficients estimation

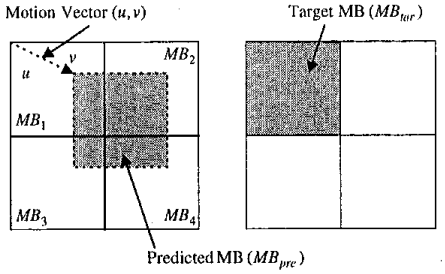


Figure 3: Graphical representation of four neighbouring MBs and motion vectors

The extraction of DCT coefficients from intra-coded frames are trivial. However, the extraction of DCT coefficients from inter-coded frames (P and B) are not trivial since these frames are motion compensated. Thus, this section explains how DCT coefficients are estimated for inter-coded frames in MPEG-2 video. Figure 4 shows four neighbouring MBs and motion vectors of the predicted (virtual) MB. DCT coefficients in inter-coded frames can be extracted from an intra-coded frame. The computation of the DCT coefficients of a new arbitrary position image block from the DCT coefficients of the four original neighbouring blocks takes the form of pre-multiplying and post-multiplying of those

blocks with appropriate matrices (Γ_{i1} and Γ_{i2}) [8]. Thus Equation (5) describes the process of DCT coefficients evaluation for a motion-compensated MB.

$$DCT(MB_{pre}) = \sum_{i=1}^{4} \xi_i \, DCT(MB_i) \, \zeta_i \tag{5}$$

where, $\xi_i = DCT(\Gamma_{i1})$ and $\zeta_i = DCT(\Gamma_{i2})$.

There are four possible locations of the sub-block of interest (with reference to MB_i): lower-right, lower-left, upper-right and upper-left. These locations define Γ_{i1} and Γ_{i2} matrices as tabulated in Table 1. Parameters h_i and w_i are the height and width of the overlap of MB_{pre} with MB_i. For a particular MB, these two parameters can be evaluated from its motion vector (u,v). Therefore, DCT coefficients of the predicted MB can be evaluated using Equation (5). DCT coefficients of the error term (MB_{er}) are readily available for MPEG-2 compressed video. Finally, DCT coefficients of the target (current) MB (MB_{tag}) are calculated by adding the DCT coefficients of the predicted MB and the DCT coefficients of the displaced frame difference (DFD) signal as shown in Equation (6).

$$\begin{aligned} DCT(MB_{tag}) &= DCT(MB_{pre} + MB_{er}) \\ &= DCT(MB_{pre}) + DCT(MB_{er}) \end{aligned} \tag{6}$$

Sub-block	Position	Γ_{i1}	Γ_{i2}
MB_1	lower right	$\begin{bmatrix} 0 & I_{h_1} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{w_1} & 0 \end{bmatrix}$
MB_2	lower left	$\begin{bmatrix} 0 & I_{h_2} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_{w_2} \\ 0 & 0 \end{bmatrix}$
MB_3	upper right	$\begin{bmatrix} 0 & 0 \\ I_{h_3} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{w_3} & 0 \end{bmatrix}$
MB_4	upper left	$\begin{bmatrix} 0 & 0 \\ I_{h_4} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_{w_4} \\ 0 & 0 \end{bmatrix}$

Table 1: Γ_{i1} and Γ_{i2} matrices
(I is an identity matrix of h_i or w_i)

4. PROPOSED VIDEO EDITING SYSTEM

Normally, motion estimation process bares 60-70% of computational complexity of a typical MPEG video encoder. Therefore, in order to keep the computational complexity of the video editing process at a reasonable level, it is desirable to reuse the motion vectors as much as possible without re-computing them. Existing motion vectors can be considered as a good approximation for motion vectors for the faded, dissolved and wiped sequences. Especially for wipe editing, existing motion vectors can be used very effectively. For fade and dissolve editing, the approximation becomes ineffective when the length of the fade or dissolve region is low. When the fade or dissolve length is low, there are significant changes between two consecutive frames and hence existing motion vectors fail to provide reasonable results. Therefore, in this proposed scheme, existing motion vectors are used together with DCT coefficients to produce these special effects in compressed video. Following sub-sections describe how this approach is applied to fading, dissolving and wiping separately.

4.1 Fade editing

DCT coefficients for every frame can be extracted as explained in section 3.3 and fade-in and fade-out operation are applied to each MB separately in DCT domain. This is explained in Equation (7) and Equation (8) respectively. After the fading operation in DCT domain, all MBs in each frame can be intra-coded. But, if all MBs in every frame are intra-coded then, the quantisation step size will be increased to maintain the same channel bit rate in a constant bit rate (CBR) system. This will degrade the performance since no correlation is exploited between frames (no motion compensation).

$$S_{n,m_1,m_2}(k_1,k_2) = \begin{cases} \left[1-\left(\frac{n-L_1}{L}\right)\right]C + \left(\frac{n-L_1}{L}\right)G_{n,m_1,m_2}(0,0) & k_1,k_2 = 0 \\ \left(\frac{n-L_1}{L}\right)G_{n,m_1,m_2}(k_1,k_2) & else \end{cases} \tag{7}$$

$$S_{n,m_1,m_2}(k_1,k_2) = \begin{cases} \left[1-\left(\frac{n-L_1}{L}\right)\right]F_{n,m_1,m_2}(0,0) + \left(\frac{n-L_1}{L}\right)C & k_1,k_2 = 0 \\ \left(\frac{n-L_1}{L}\right)F_{n,m_1,m_2}(k_1,k_2) & else \end{cases} \tag{8}$$

where, $S_{n,m_1,m_2}(k_1,k_2)$ is DCT coefficients of $s_{n,m_1,m_2}(x,y)$, $F_{n,m_1,m_2}(k_1,k_2)$ is DCT coefficients of $f_{n,m_1,m_2}(x,y)$, $G_{n,m_1,m_2}(k_1,k_2)$ is DCT coefficients of $g_{n,m_1,m_2}(x,y)$, m_1, m_2 is location of the MB in the image, k_1,k_2 are location in the MB ($k_1,k_2 = 0:7$).

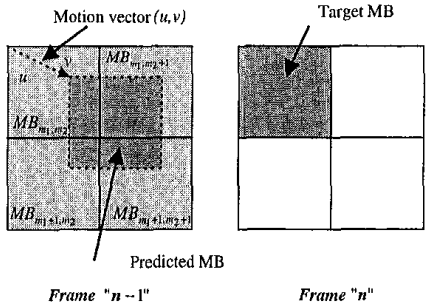


Figure 5: Graphical representation of four neighbouring MBs and motion vectors (after fading)

$$DFD^{new}_{n,m_1,m_2}(k_1,k_2) = S_{n,m_1,m_2}(k_1,k_2) - \sum_{i=1}^{4} \xi_i \Phi_i \zeta_i \tag{9}$$

Motion vector	Φ_i
$u \geq 0, v \geq 0$	$\Phi_1 = S_{n-1,m_1,m_2}(k_1,k_2), \Phi_2 = S_{n-1,m_1,m_2+1}(k_1,k_2),$ $\Phi_3 = S_{n-1,m_1+1,m_2}(k_1,k_2), \Phi_4 = S_{n-1,m_1+1,m_2+1}(k_1,k_2),$
$u < 0, v \geq 0$	$\Phi_1 = S_{n-1,m_1,m_2-1}(k_1,k_2), \Phi_2 = S_{n-1,m_1,m_2}(k_1,k_2),$ $\Phi_3 = S_{n-1,m_1+1,m_2-1}(k_1,k_2), \Phi_4 = S_{n-1,m_1+1,m_2}(k_1,k_2),$
$u \geq 0, v < 0$	$\Phi_1 = S_{n-1,m_1-1,m_2}(k_1,k_2), \Phi_2 = S_{n-1,m_1-1,m_2+1}(k_1,k_2),$ $\Phi_3 = S_{n-1,m_1,m_2}(k_1,k_2), \Phi_4 = S_{n-1,m_1,m_2+1}(k_1,k_2),$
$u < 0, v < 0$	$\Phi_1 = S_{n-1,m_1-1,m_2-1}(k_1,k_2), \Phi_2 = S_{n-1,m_1-1,m_2}(k_1,k_2),$ $\Phi_3 = S_{n-1,m_1,m_2-1}(k_1,k_2), \Phi_4 = S_{n-1,m_1,m_2}(k_1,k_2),$

Table 2: Definition of Φ_i

where, $DFD^{new}_{n,m1,m2}(k_1,k_2)$ is new DFD signal (after fading) for (m_1, m_2) MB in n^{th} frame.

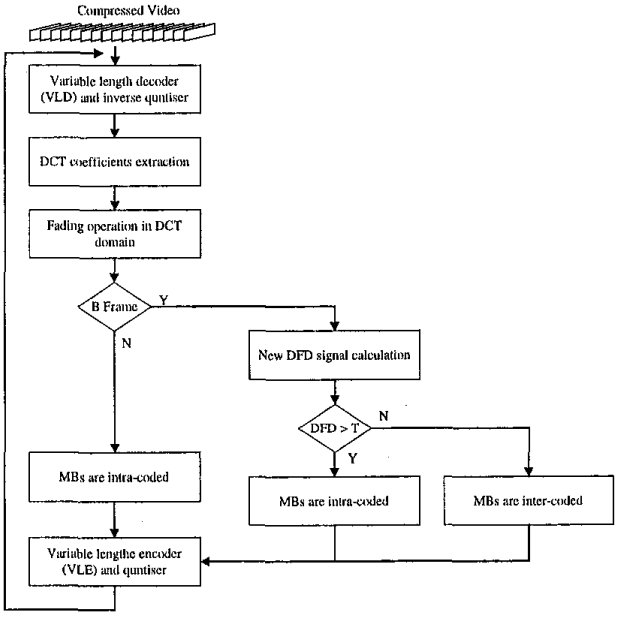


Figure 6: Complete algorithm for fading operation in compressed domain

In this proposed scheme existing motion vectors are used and corresponding DFD signals are modified accordingly. New DFD signal calculation for fading is clearly illustrated in Figure 5 and Equation 9. Parameter Φ_i is dependent on the sign of the motion vector and table 2 presents the parameter Φ_i for all possible combinations. Thus, the DFD signals are re-calculated with the existing motion vectors and all MBs are coded as before, unless DFD signal is very large. If the DFD signal exceeds a predetermined threshold (T) then this particular MB needs to be intra-coded. The length of these fading operations can be controlled by the user, typically between 30-80 frames. Since existing motion vectors are not optimised for the faded sequence, new DFD signal is comparatively large with respect to the old DFD signal for a small fade length (L). However, new DFD signal reaches to its previous DFD signal when fade length (L) increases. Therefore in general, if inter-coded frames are used continuously, picture quality may be degraded (especially high frequency components) as same quantisation matrices are used to quantise new DFD signals. To avoid this situation, intra-coded frames are used periodically to maintain the required picture quality. Therefore, in this proposed scheme existing motion vectors are used for all B-frames and all MBs in P-frames are intra-coded. This will guarantee to avoid error propagation since B-frames are not used for any predictions. Figure 6 presents the complete algorithm for fading.

4.2 Dissolve editing

As explained in section 4.1, dissolving can also be done in DCT domain in a similar fashion. This process is explained in Equation (11).

$$S_{n,m_1,m_2}(k_1,k_2) = \left[1-\left(\frac{n-L_1}{F}\right)\right]F_{n,m_1,m_2}(k_1,k_2) + \left(\frac{n-L_1}{F}\right)G_{n,m_1,m_2}(k_1,k_2) \tag{11}$$

Same algorithm can also be applied for dissolving as well. However, for dissolved sequence there are two sets of motion vectors from encoder-1 (f_n) and encoder-2 (g_n). Thus, two new DFD signals are evaluated with two sets of motion vectors and the lowest DFD signal should provide the best motion vector for a particular MB from available existing motion vectors. Therefore, new DFD signal is evaluated according to the Equation (12) for all B-frames and motion vectors are modified accordingly. Apart from this modification, same algorithm is applied for dissolve editing as well.

$$DFD^{new}_{n,m}(k_1,k_2) = Min\left\{\begin{matrix}\left[S_{n,m_1,m_2}(k_1,k_2) - \sum_{i=1}^{4}\xi_i\Phi_i\zeta_i\right]_{MV_{encoder1}}, \\ \left[S_{n,m_1,m_2}(k_1,k_2) - \sum_{i=1}^{4}\xi_i\Phi_i\zeta_i\right]_{MV_{encoder2}}\end{matrix}\right\} \tag{12}$$

4.3 Wipe editing

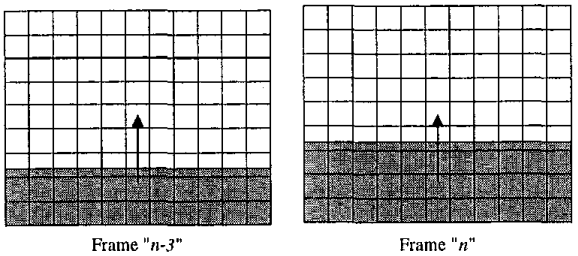


Figure 7-a: Vertical wiping with picture F and G in uncompressed domain (Frame *"n-3"* and Frame *"n"*)

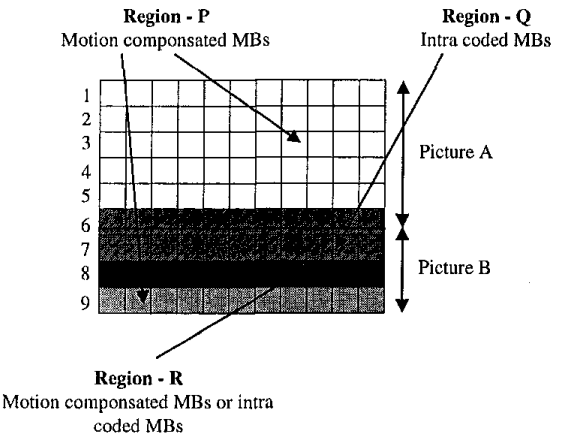


Figure 7-b: Vertical wiping with picture F and G in compressed domain (Frame "*n*")

During a wipe transition changes occur along the transition boundary. Therefore, existing motion vectors can be used very effectively for all MBs other than MBs on and close to the transition boundary. Thus for wipe editing, all MBs on the transition region are intra-coded and all other MBs are motion compensated using the existing motion vectors.

Forward vertical wiping is considered to explain the algorithm. There each small rectangle represents a four 8x8 MBs. An intermediate step of this wiping operation is shown in Figure 7-a. Assume that frame "*n-3*" is an I/P frame whereas frame "*n*" is a P-frame. Figure 7-b presents how MBs are coded in compressed domain for this wiping pattern. Rows one to five and nine (**region-P**) in frame "*n*" can be motion compensated from two sequences *F* and *G* (frame "*n-3*" and frame "*n*") respectively. MBs in row eight (**region-R**) either can be intra-coded or motion compensated depending on the magnitude of motion vectors. If any motion vector in region-R points to row seven, these MBs are intra-coded. Otherwise MBs in row seven are motion compensated. Rows six and seven (**region-Q**) are intra-coded as they cannot be motion compensated. Motion vectors and relevant DFD signals are extracted from the two sequences accordingly. DCT coefficients are calculated using the technique explained in section 3.3. It should be noted that MBs in row six are composed with both picture F and picture G. DCT coefficients of these MBs can be calculated as shown in Equation (13). Similar set of relationships can be derived for other wiping patterns as well. Table 3 shows some of these relationships for common wiping patterns.

$$DCT(MB_{resultant}) = DCT(\zeta)DCT(MB_A) + DCT(\vartheta)DCT(MB_B) \quad (13)$$

where,

$$\phi = \begin{bmatrix} I_{y_i} & 0 \\ 0 & 0 \end{bmatrix}, \qquad \vartheta = \begin{bmatrix} 0 & 0 \\ 0 & I_{8-y_i} \end{bmatrix}$$

Parameter y_i is dependent on wiping speed, which is selected by the user.

Wiping pattern	$DCT(MB_{resultant})$
Vertical	$DCT(\phi)DCT(MB_A) + DCT(\vartheta)DCT(MB_B)$
Horizontal	$DCT(MB_F)DCT(\phi) + DCT(MB_G)DCT(\vartheta)$
Barn-door (Vertical)	$DCT(\phi_1)DCT(MB_F) + DCT(\vartheta_1)DCT(MB_G)$
Barn-door (Horizontal)	$DCT(MB_F)DCT(\phi_1) + DCT(MB_G)DCT(\vartheta_1)$
Box	$DCT(\phi_1)DCT(MB_F)DCT(\phi_1) +$ $DCT(\vartheta_1)DCT(MB_G) +$ $DCT(\phi_1)DCT(MB_F)DCT(\vartheta_1)$

Table 3: Relationships for common wiping patterns

where $\phi = \begin{bmatrix} I_{y_i} & 0 \\ 0 & 0 \end{bmatrix}$, $\vartheta = \begin{bmatrix} 0 & 0 \\ 0 & I_{8-y_i} \end{bmatrix}$, $\phi_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & I_{y_i} & 0 \\ 0 & 0 & 0 \end{bmatrix}$,

$$\vartheta_1' = \begin{bmatrix} I_{x_i} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{8-y_i-x_i} \end{bmatrix}$$

Similar arguments can be followed for other frames as well. Therefore, forward vertical wiping can be produced in compressed domain with this approach. Vertical backward, horizontal forward, horizontal backward, barn-door forward, barn-door backward, box-wipe backward, box-wipe forward can also be generated in a similar fashion.

4.4 Implementation of the proposed video editing system

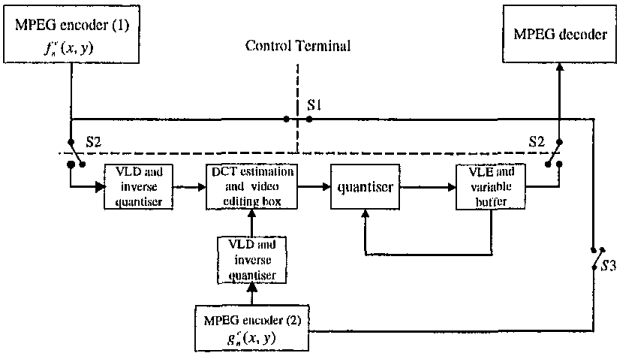


Figure 8: Proposed compressed domain video editor

Figure 8 shows the proposed compressed domain video editor. Two MPEG compressed video streams are applied to the editor and these two streams are partially decoded with variable length decoder (VLD) and the inverse quantiser.

With this data, DCT coefficients are estimated for each frame as explained in section 3.3. Then these coefficients and motion vectors are applied to the video editing box to process the parameters as explained previously. Finally, all MBs are quantised to suit the channel bit rate. The quantisation is achieved through the feedback loop from the buffer as in normal MPEG-2 video encoder.

When user needs to switch the video into a different sequence through any special effects operation, S2 switch will be closed and S1 switch will be opened through control terminal. There are M locations, which fading/dissolving can be terminated within a GOP (M,N) structure. Therefore, proposed algorithm has made allowances for various end points in the MPEG sequence (i.e. I, P or B frames). It should be noted that the last frame in the fade-in/dissolving operation is an unaltered picture from the second sequence, $g_n(x,y)$. Therefore, if the last frame of the fade-in/dissolving process is an I-picture or a P-picture (originally) as shown in Figure 9-a (M=12, N=3), then no further processing is required for following frames to synchronise with the sequence. However for fade-out process last frame is a solid colour picture. Therefore, following P-frame has to be intra-coded and B1 and B2 frames need to be intra-coded unless any MBs are backward predicted in the original sequence. This criteria is presented in Figure 9-b. If the last frame is a B1- picture (Figure 9-c), following P-frame need to be intra-coded and B2 frame is intra-coded or backward predicted as in the previous case. If B2-picture is the last frame of fading and dissolving (Figure 9-d), following P-frame has to be inta-coded. This is required since forward motion vectors for B1 and B2 frames are incorrect due to the changes in the previous reference frame (due to fading or dissolving). After the wipe operation, S2 switch will be opened and S3 switch will be closed.

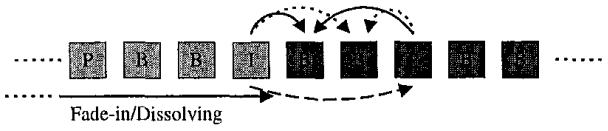


Figure 9-a: Fade-in/dissolving ends at I/P-frame

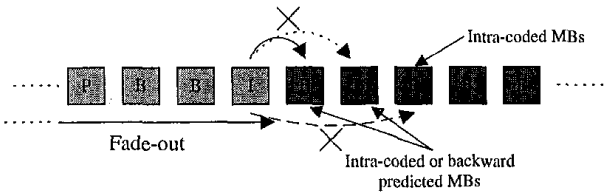


Figure 9-b: Fade-out ends at I/P-frame

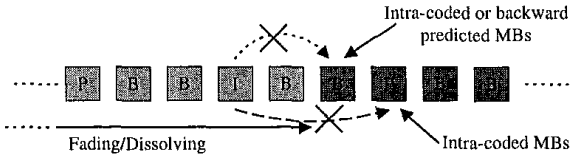


Figure 9-c: Fading/dissolving ends at first B-frame

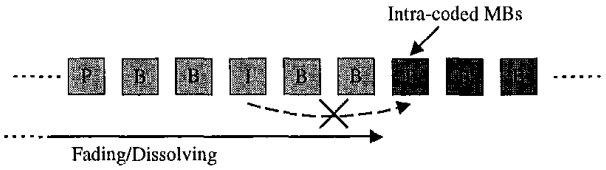


Figure 9-d: Fading/dissolving ends at second B-frame

5. RESULTS

Video special effects editing using the proposed scheme are considered here. MPEG-2 video bit streams are used as input to the proposed compressed domain video editor. Results are compared with the conventional scheme (Figure 3). PSNR of the luminance signal is considered to measure the objective quality of the edited video using these techniques. Signal $s_n(x,y)$ (output at the uncompressed video editor – Figure 2) is considered as the reference signal for all PSNR calculations.

5.1 Fade-in and Fade-out

Fade-in and fade-out special effects editing results can also be compared with DC-only scheme [7]. Figure 10 shows the comparison for a fade-in special effect editing in compressed domain. Figure 11 shows the same for fade-out special effect editing. Results show that PSNR of the DC-only scheme [7] is very low. PSNR of the DC-only scheme is decreasing, until it finds the next I-frame. Even at I-frames PSNR is well below the expected value. But, PSNR of the proposed scheme closely follows the PSNR of the conventional scheme. PSNR of the proposed scheme is very close to the conventional scheme at I-frames. Experimental results showed that PSNR of the DC-only scheme is heavily dependent on the solid colour and the nature of the MPEG-2 video sequences. However, the proposed scheme is independent of these parameters. Figure 10 also shows that PSNR of the DC-only scheme is very low even after the fade-in operation, until it finds the next I-frame. This is due to accumulated error in the last frame of the fade-in sequence and the failure to provide a solution to end point synchronisation. This problem cannot be seen in Figure 11 since fade-out ends exactly at the last frame of a GOP. In the proposed scheme, algorithm made allowances for various end points in the MPEG sequence (i.e. I, P or B frames). Figure 12 shows the subjective quality of the 70th frame of fade-in operation. It is clear that subjective quality of the DC-only method is poor compared to the other two schemes.

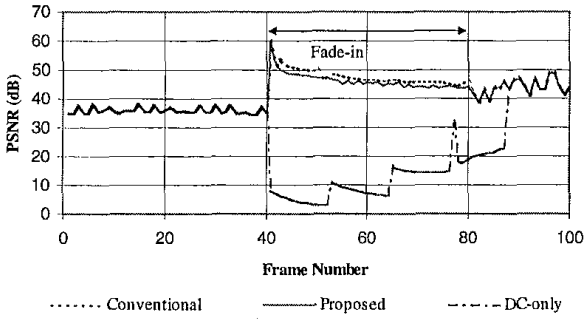


Figure 10: Performance comparison for fade-in production

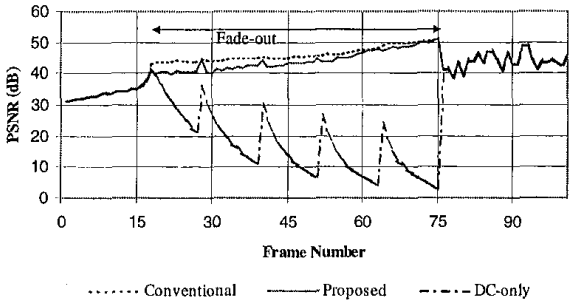


Figure 11: Performance comparison for fade-out production

Figure 12: 70th frame of the fade-in operation
(from left, conventional, proposed and DC-only)

Conventional scheme is computationally very expensive compared to the proposed scheme. Although DC-only method is less computationally expensive, the quality of the edited video is poor. Performance of the fade editing algorithm is also tested with different parameter settings. Following sections summarise these performance.

- **PSNR performance against M**

Figure 13 presents PSNR variation against the number of frames in the GOP structure (M). It shows that rate of fall in PSNR for conventional scheme is higher than that of the proposed scheme. Since correlation between frames is less during a fading operation, frequent intra-frames help to reduce the DFD signal in conventional scheme. As the span between two intra-frame increases, the DFD signal increases and reaches the proposed scheme. Therefore, the difference in PSNR between the conventional and the proposed scheme reduces with the length of the GOP structure.

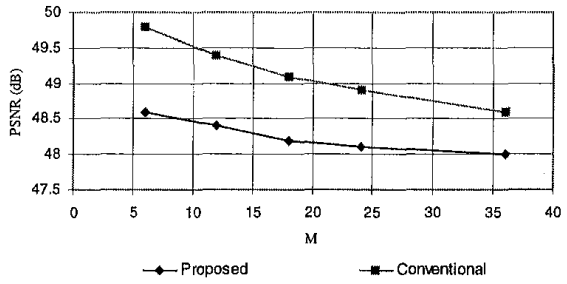


Figure 13: PSNR comparison with the number of frames in a GOP structure (N=3, fade length, L=40)

- **PSNR performance against N**

Figure 14 shows the PSNR comparison with the I/P frame distance (N). Since existing motion vectors are not good enough to consider as the faded sequence motion vectors for large values of N, PSNR of the proposed scheme reduces with N. Conventional scheme is optimised only for a particular value of N. In this example N=2 gives the optimum results. In conventional scheme PSNR is reducing for higher values of N as well.

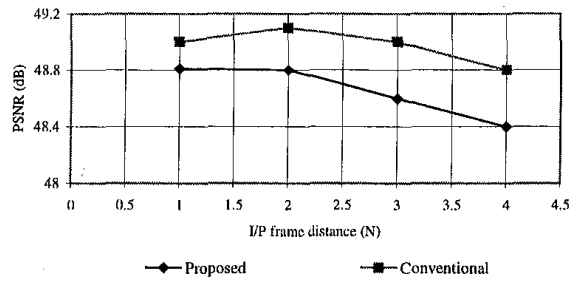


Figure 14: PSNR comparison with the I/P frame distance
(M=12, fade length, L=40)

- **PSNR performance against fade length (L)**

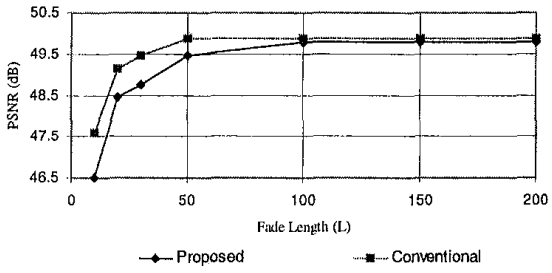


Figure 15: Performance comparison with the fade length
(M=12, N=3)

Figure 15 presents the PSNR variation for different fade lengths. As explained in section 4.1 performance of the proposed scheme is similar to the performance of the conventional scheme when the fade length is increased. When fade length increases few changes are happening during consecutive frames in the faded sequence and existing motion vectors produces better results.

5.2 Dissolving

Figure 16 shows the comparison for a dissolve special effect editing in compressed domain. It describes that PSNR of the proposed scheme closely follows the PSNR of the conventional scheme. Figure 17 shows the subjective quality of the 28th frame of dissolve operation. Subjective quality of the proposed scheme is again similar to the conventional scheme. Results shows that drop in average PSNR for dissolve editing is slightly higher compared to the same for fading. This is expected since the accuracy of motion compensation is higher when one sequence is only a solid colour (during fading).

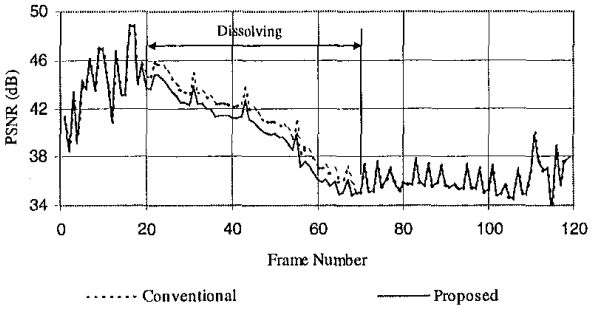


Figure 16: Performance comparison for dissolve production

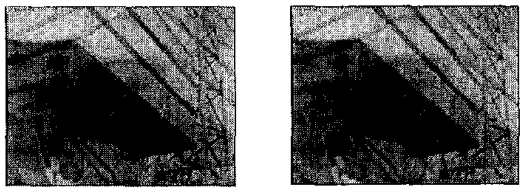


Figure 17: 28th frame of the dissolve editing *(from left, conventional and proposed)*

Figure 18 presents the PSNR variation with different dissolve lengths. As in the fading, performance of the proposed scheme achieves the performance of the conventional scheme when the dissolve length increases. As in the fade editing, similar observations can be made with different M and N values for dissolve editing as well.

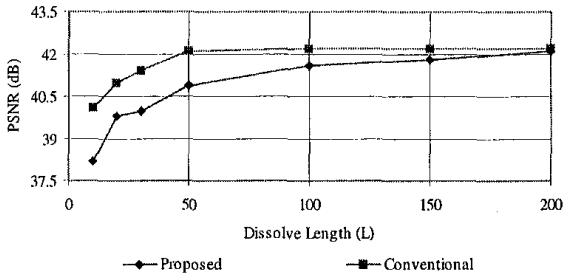


Figure 18: Performance comparison for dissolve editing with the dissolve length *(M=12, N=3)*

5.3 Wiping

Figure 19 shows the comparison for a forward vertical wiping special effect editing in compressed domain. It shows that PSNR of the proposed scheme closely follows the PSNR of the conventional scheme. Unlike in fading and dissolving, the drop in average PSNR is very small. This is due to existing motion vectors can be used very effectively in wipe editing. Performance comparison of the subjective quality of the wipe edited video is presented in Figure 20. Subjective quality of the proposed scheme is again similar to the conventional scheme.

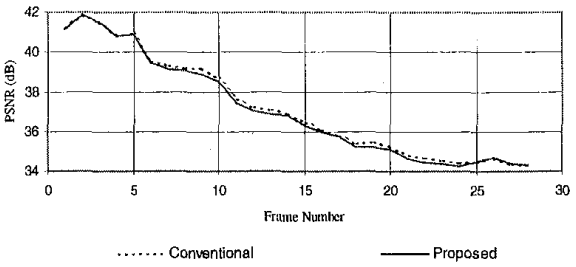


Figure 19: Performance comparison for wipe production

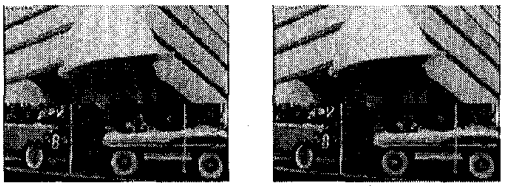


Figure 20: 12th frame of the wipe editing *(from left, conventional and proposed)*

Figure 21 presents the performance comparison of the two schemes with different wipe lengths. Unlike in fading and dissolving, performance of the proposed wipe editing algorithm does not approach the performance of the conventional scheme with the wipe length (L). Average number of intra-coded MBs are increased for large wipe length and this generate a feed back to increase the quantisation step size in order to maintain the same bit rate. However the drop in PSNR is very small.

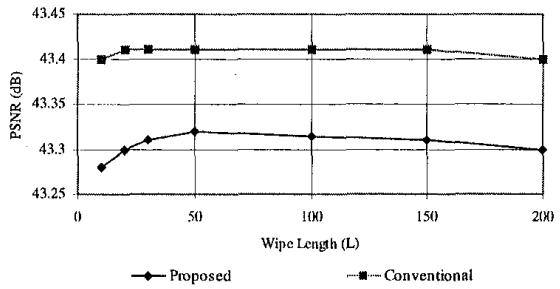


Figure 21: Performance comparison for wipe editing with the wipe length *(M=12, N=3)*

Description	Number of Frames	Average PSNR (dB)	
		Conventi -onal	Proposed
Forward Vertical Wiping	10	41.0	40.2
Forward Horizontal Wiping	12	40.1	39.6
Backward Barn-doors Wiping	20	37.6	37.1
Forward Horizontal Wiping	20	47.8	47.4
Backward Vertical Wiping	30	47.4	46.8
Forward Horizontal Wiping	10	45.8	45.2
Forward Barn-doors Wiping	15	48.4	47.7
Dissolve	50	41.0	39.8
Dissolve	60	40.1	38.9
Dissolve	30	37.6	36.5
Fade-out	40	47.8	46.4
Fade-out	50	47.4	46.6
Fade-in	50	45.8	44.9
Fade-in	30	48.4	47.4
Fade-in	20	48.2	47.3

Table 4: Summarised results

We tested the proposed scheme with different fade, dissolve, wipe rates and different MPEG-2 video bit streams and observed similar results. Some of these results are summarised in Table (4). The drop in average PSNR of the proposed scheme (for fade, dissolve and wipe) is acceptable when the reduction in computational complexity is taken in to account. Conventional scheme is computationally very expensive compared to the proposed scheme and make it difficult for real time implementations. Therefore, these special effects editing can be done in compressed domain without full frame decompression and motion re-estimation.

6. CONCLUSIONS

In this paper, a novel algorithm is proposed for fade-in, fade-out, dissolve and wipe video editing operations in MPEG-2 compressed domain without full frame decompression and motion re-estimation. DCT coefficients are estimated for all frames and used these DCT coefficients together with the existing motion vectors to produce these special effects in compressed domain. Results show that the subjective and objective quality of the edited video closely follows the quality of the conventional method. Unlike the conventional scheme, proposed scheme is computationally inexpensive and makes it possible for real time implementations. Future work is required to extend this work for complex video special effects.

REFERENCES

1. Sikora, T., *"MPEG Digital Video-Coding Standards,"* IEEE Signal Processing Magazine, pp. 82-100, September 1997.

2. ITU-T, *"Draft H.263,"* May 1996.

3. Wallace, G.K., *" The JPEG Still Picture Compression Standard,"* Communications of the ACM, Vol. 34, pp. 30-44, April 1991.

4. Shen, B., Sethi, I.K., *" Inner-Block Operations on Compressed Images,"* Proceedings of ACM Multimedia Conference, San Francisco, California, USA, 1995.

5. Shen, B., Sethi, I.K., Bhaskaran, V.,*" DCT Domain Alpha Bending,"* Proceedings of International Conference on Image Processing, Vol. 1, Ch. 205, pp.857-861, 1998.

6. Smith, B.C., Rowe, L.A., *"Algorithms for Manipulating Compressed Images,"* IEEE Computer Graphics and Applications, pp. 34-42, September 1993.

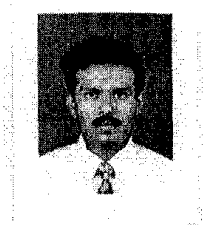
7. Shen, B., *" Fast Fade-out Operation on MPEG Video,"* ICIP, CD Paper Number - MP10.01, 1998.

8. Chang, S.F., Messerschmitt, D.G., *" Manipulation and Composting of MC-DCT Compressed Video,"* IEEE Journal of Selected Areas in Communications, vol.13, pp.1-11, January 1995.

ACKNOWLEDGEMENTS

First author would like to express his gratitude and sincere appreciation to the University of Bristol and CVCP for providing financial support for this work.

BIOGRAPHIES



Anil Fernando received the B.Sc. Engineering degree (First class) in Electronic and Telecommunications Engineering from University of Moratuwa, Sri Lanka in 1995 and the MEng degree (Distinction) in Telecommunications from Asian Institute of Technology, Bangkok, Thailand in 1997. Since 1998, he has been a Ph.D. student at Department of Electrical and

Electronic Engineering, University of Bristol. His current research interests include scene change detection in uncompressed and compressed video, video editing in compressed video, intelligent video encoding, COFDM for wireless channels, channel coding and modulation schemes for satellite channels.

Nishan Canagarajah is currently a Senior Lecturer in Signal Processing at University of Bristol. He has BA (Hons) and Ph.D. in DSP techniques for speech enhancement, both from the University of Cambridge. He is a committee member of the IEE Professional Group E5, member of the virtual centre of excellence in digital broadcasting and multimedia technology and an associate editor of the IEE Electronics and Communication Journal. He is also an editor of a book on Mobile Multimedia Technology. His research interests include image and video coding, non-linear filtering techniques and the application of signal processing to audio and medical electronics.

David Bull is currently a Professor of Digital Signal Processing and a University of Bristol research Fellow. He leads the Image Communications Group at Bristol and is Deputy Director of the Centre for Communications Research. He has worked widely in the fields of 1 and 2-D signal processing and his current research is focused on the problems of image and video communications for both low bit rate and broadcast applications. In particular he is working on error resilient source coding, linear and non-linear filter banks, scalable coding methods, motion estimation and architectural optimisation (for filters, transforms and wavelet filter banks).